

## 25

## 2

This can occur for a number of reasons. For example, the dictionary may be created based upon a subset of interrogated data, which turns out to be a poor representative of the database as a whole. Or, in the case of a "static" dictionary, the

dictionary may be accurate when created, but the nature of the data changes, causing the dictionary to become stale. This phenomenon frequently occurs with time-dependent data. The static dictionary may be created, for example, in February, when much of the underlying data includes the word "February". When March arrives, most of the underlying data include the word "March" rather than "February"; since the original static dictionary does not contain "March", its compression activities are poorly guided with respect to the current data sought to be compressed.

As a result of the foregoing conditions, certain data cannot be compressed because the data is missing from the dictionary. Yet, compression is still attempted for this data, albeit unsuccessfully. And these frustrated "compression calls" still require time to perform, occupying valuable processor time, which could otherwise be spent performing other tasks.

One option is to simply deactivate compression. Although this approach saves processor time otherwise spent on frustrated compression attempts, the input/output efficiency suffers because data is now stored in its full-length, uncompressed form. Another option is to rebuild the dictionary anew, in accordance with the current data. Rebuilding the dictionary, however, typically requires taking the data storage subsystem off-line. This is simply not an option for certain applications, where continuous data availability is crucial, such as automated teller machines, internationally accessible financial data, twenty-four hour telephone directory services, etc.

Consequently, known compression schemes are not completely adequate for some applications due to certain unsolved problems.

### SUMMARY OF THE INVENTION

Broadly, the present invention concerns the on-line analysis of data compression efficacy in a data storage system, where data compression is modified in accordance with the results of such analysis. In one embodiment, multiple input data blocks are received for storage in a data storage subsystem. A predetermined compression process is applied to the data blocks. Application of the predetermined

5

10

20

## BRIEF DESCRIPTION OF THE DRAWINGS

25

4

**FIGURE 3** is a perspective view of an exemplary signal-bearing medium embodying a data storage medium in accordance with the invention.

**FIGURE 4** is a basic flowchart illustrating data compression according to the invention.

**FIGURES 5A-5B** are a detailed flowchart illustrating a specific example of data compression according to the invention.

**FIGURE 6** is a flowchart illustrating an example of operational steps performed after detecting poor compression efficacy.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

### HARDWARE COMPONENTS & INTERCONNECTIONS

One aspect of the invention concerns a data storage subsystem that stores data with data compression, analyzes the compression to provide feedback, and regulates compression according to the feedback. The data storage subsystem of the invention may be embodied by various hardware components and interconnections. To show an example, Figure 1 depicts a data storage subsystem 100. The subsystem 100 includes a host 102, storage controller 104, and storage unit 106.

#### Storage Unit

Preferably, the storage unit 106 comprises a cost efficient data storage unit suitable for a data storage subsystem. In this respect, the storage unit 106 may comprise a direct access storage device (DASD) such as one or more magnetic hard disk drive assemblies. In addition, or as an alternative, other storage means may be employed, such as magnetic tape storage, optical tape, optical disks, integrated circuit memory, or any other data storage type.

Since the storage unit 106 is cost efficient, this is a preferable location for a data dictionary <sup>118</sup>~~116~~, rather than storing the dictionary <sup>118</sup>~~116~~ elsewhere in more expensive storage, such as random access memory. However, cache 112 is preferably used to store as much of the dictionary as space permits, to minimize dictionary access

00866507-153097

20

25

5

A time. The data dictionary <sup>118</sup>~~116~~ comprises a static data dictionary, which may be formed using a novel technique or by one or many techniques familiar to ordinarily skilled artisans.

### Controller

5 The storage controller 104 provides an interface between the host 102 and the storage unit 106. The controller 104 may comprise a new component satisfying this purpose, or an existing product depending upon the needs of the application.

### Host: Generally

10 The host 102 comprises a computer performing an application program requiring storage and retrieval of data on the storage unit 106. The host 102 may comprise any digital data processing machine, specific examples of which include mainframe computers, personal computers, workstations, etc.

15 The host 102 is connected to the controller 104 via an input/output line 111, which may comprise one or more wires, cables, fiber optic links, electromagnetic links, intelligent channels, or any other suitable means for conveying data and commands between the host 102 and controller 104. The host 102 includes a number of components, which may be constructed using suitable hardware or software implementations, as shown below. These components include a compression unit 116, an output buffer 108, an input buffer 110, a dictionary cache 112, and a processing unit 114.

*Host*  
B Host: Buffers

20 The input buffer 110 is coupled to the compression unit 116. The input buffer 110 receives data blocks provided by the host 102 for storage in the unit 106, and makes the input data blocks available to the compression unit 116. Although the input buffer 110 may be implemented by suitable memory hardware such as random access memory, it may also comprise an appropriate software construct.

25

6

The output buffer 108 is coupled to the compression unit 116 and the processing unit 114. Input data blocks from the input buffer 110 that are compressed are stored by the compression unit 116 in the output buffer 108. In some cases, as explained below, data blocks are routed to the output buffer 108 without any compression. Although the input buffer 110 may be implemented by suitable memory hardware such as random access memory, it may also comprise an appropriate software construct.

#### Host: Dictionary Cache

The dictionary cache 112 is coupled to the compression unit 116. The dictionary cache 112 contains a cached portion of the static dictionary <sup>112</sup>~~116~~, enabling the compression unit 116 to more quickly compress data from the input buffer 110 according to the established dictionary. The dictionary cache 112 may be implemented by suitable memory hardware such as random access memory, or an appropriate software construct if desired. To speed operation of the compression unit 116, the dictionary cache 112 is preferably as large as expenses permit, and can contain the entire dictionary if desired.

#### Host Processing Unit

The processing unit 114 manages transfers of compressed data from the output buffer 108 to the storage controller 104 and eventually to the storage unit 106. The processing unit 114 may also manage retrieval of data from the storage unit 106 via the storage controller 104. The processing unit 114 may comprise one or more microprocessors, a personal or mainframe computer, custom integrated circuit, discrete circuitry, or any other electronic or software module to manage exchanges of data between the storage unit 106 and the output buffer 108. This may also include, for example, other tasks such as addressing and formatting of data.

Host: Compression Unit

A 118 The compression unit 116 applies the compression scheme of the dictionary  
116 to input data received into the input buffer 110, sending the compressed data to  
the output buffer 108. As mentioned above, transfers of compressed data from the  
5 output buffer 108 to the storage unit 106 are conducted by the processing unit 114.  
In accordance with the invention, the compression unit 116 also evaluates the success  
of its compression, and regulates future compression in accordance with these  
evaluations. These tasks are discussed in much greater detail below.

10 The compression unit 116 may be implemented in various ways, depending  
upon the specific needs of the application. In one embodiment, the compression unit  
116 may be implemented by an electronic module including discrete circuitry,  
programmable circuit components, logic circuitry, or a combination. Preferably, the  
compression unit 116 is embodied by suitable application-specific integrated circuitry.  
As another alternative, some of the features of this module may be implemented in  
software, while others are implemented using suitable hardware.

20 In another embodiment, the compression unit is implemented completely in  
software, by using a digital computer to execute a sequence of programming  
instructions. In this embodiment, then, the components of the formatting unit  
comprise software modules or functional units, rather than actual hardware  
components. This embodiment is therefore implemented using a digital computer 200,  
as shown in Figure 2.

25 The computer 200 includes a processing unit 202, such as a microprocessor or  
other processing machine, coupled to an input/output line 208. The processing unit  
202 is also coupled to a storage unit 203, an example of which includes a fast-access  
storage unit 204 and a nonvolatile storage (NVS) unit 206. The fast-access storage  
unit 204 preferably comprises random access memory, and may be used to store the  
programming instructions executed by the processing unit 202. The nonvolatile  
storage unit 206 may comprise, for example, one or more magnetic data storage disks  
such as one or more "hard drives", tape drives, or other suitable storage devices. The

5

10

15

20

## 25

9

This signal-bearing media may comprise, for example, RAM (not shown) contained within or otherwise accessible to the compression unit 116. Alternatively, the instructions may be contained in another signal-bearing media, such as a magnetic data storage diskette 300 (Figure 3), directly or indirectly accessible by the compression unit 116. Whether contained in the host 102 or elsewhere, the

instructions may stored on a variety of machine-readable data storage media, such as DASD storage (e.g., a conventional "hard drive" or a RAID array), magnetic tape, electronic read-only memory (e.g., CD-ROM or WORM), an optical storage device (e.g. WORM), paper "punch" cards, or other suitable signal-bearing media including transmission media such as digital and analog and communication links and wireless. In an illustrative embodiment of the invention, the machine-readable instructions may comprise lines of compiled PLX or PLAS language code.

#### General Sequence of Operation

Figure 4 depicts an sequence 400 for data compression in accordance with the invention. As explained more fully below, all of the steps 400 may be performed by the subsystem 100, or some may be shared with a human operator (not shown). For ease of explanation, but without any limitation intended thereby, the example of Figure 4 is described in the context of the data storage subsystem 100 described above. As an example, the routine 400 may be initiated or "called" in step 402 by a hierarchically superior process, referred to as the "calling process". After the routine 400 begins in step 402, the input buffer 110 receives a number of input data blocks. A data "block" preferably comprises a contiguous group of related data, such as one record, multiple records, or another convenient unit of data. As shown by the routing 405, step 404 may be performed repeatedly during the performance of subsequent steps beginning with step 406. The data blocks are preferably received in certain particular order, this order being preserved in the input buffer 110.

In step 406, the compression unit 116 applies a predetermined compression process to the next data block in the input buffer 110. The compression process may comprise a novel compression process, or another process familiar to ordinarily skilled artisans. One example is the LZ2 technique, disclosed in U.S. Patent No. 5,561,421, and assigned to International Business Machines Corporation. As shown by the routing 407, step 406 may repeat during the performance of subsequent steps beginning with step 408. The compression unit 116 stores each compressed data block in the output buffer 108, for subsequent transfer to the storage unit 106.

1005660715  
7605507099880

20

25

In step 408, the compression unit 116 evaluates compression. As explained more thoroughly below, this may be performed by evaluating the individual compression ratios of data blocks from a fixed or running window, or by evaluating the aggregate compression ratio for data blocks in a fixed or running window. After  
5 step 408, the compression unit 116 in step 410 determines whether the evaluated compression passes a predetermined criteria. If the evaluated compression passes, control is transferred to step 414, returning compressed data in the output buffer 108 to the calling process for storage in the unit 106.

On the other hand, if the evaluated compression fails, appropriate action is  
10 taken in step 412. As an example, the compression unit 116 and/or processing unit 114 may automatically disable compression, thus storing data in the storage unit 106 in uncompressed form. Disabled compression may continue indefinitely, or it may end after predetermined time or after storage of a number of uncompressed data blocks.

Alternatively, or in addition, step 412 may involve appropriate steps taken by a human operator to deal with the failed compression. For example, the human operator may simply disable compression indefinitely, adjust the parameters of compression evaluation, or initiate formation of a new dictionary. After step 412, the routine ends in step 414.

#### More Specific Examples

  
20

Figures 5A, 5B and 6 provide more specific example of one implementation of the routine 400. For ease of explanation, but without any limitation intended thereby, the examples of Figures 5A, 5B and 6 are is described in the context of the data storage subsystem 100 described above. Figures 5A-5B depict a machine-executed sequence 500, performed by the storage unit 104, as one implementation of  
25 the routine 400 (Figure 4). Figure 6 shows a sequence 600 performed by a human operator, as an example of how step 412 may be implemented.

Machine-Executed Steps: Compression & Evaluation (Figures 5A-5B)Adjustable Parameters

5 The routine 500 (Figures 5A-5B) operates according to a number of adjustable parameters. These parameters determine how compression is evaluated and regulated. Preferably, these parameters are selected by a human operator, and downloaded to the compression unit 116 for use thereby. In this respect, the parameters are programmable, and may even be changed during operation of the sequence 500. In the illustrated embodiment, the invention utilizes adjustable parameters as shown in Table 1, below. The functions of these parameters are explained in conjunction with the sequence 500 below.

Parameter	Function
GOOD COMPRESS RATIO	what % of compression is deemed "good" for an individual compress call?
NUMBER TO SAMPLE	how many compress calls are to be sampled for evaluation?
GOOD SAMPLE RATIO	what ratio of "good" to "bad" compress calls, sampled for evaluation, are acceptable?
NUMBER TO SKIP	in response to unacceptable compression, how many compress calls are to be skipped before performing compression again?

Table 1: Adjustable Parameters

Alternatively, if desired, the adjustable parameters may be fixed, without requiring any user selection of the parameters.

Counters/Indicators

20 The routine 500 also utilizes a number of counters and indicators. Unlike the adjustable parameters, the counters and indicators are not user-defined. The counters and indicators assist the compression unit 116 by recording various

statistics, which help to evaluate compression. In the illustrated embodiment, the invention utilizes counters/indicators as shown in Table 2, below. The individual functions of these counters/indicators are explained in conjunction with the sequence 500 below.

Counter	Function
WELLNESS ACTIVE	is compression evaluation enabled?
SKIP ACTIVE	in response to unacceptable compression, has the "skip mode" been activated?
GOOD COMPRESS CALLS	how many compress calls have been "good"?
NUMBER SAMPLED	how many compress calls have been sampled/evaluated?
NUMBER SKIPPED	since entering the skip mode, how many compress calls have been skipped?

**Table 2: Counters/Indicators**

Sequence 500: Step-by-Step Description

Steps 502-505 are initiated whenever the host 102 "opens" a group of multiple data blocks, called a data set. As known in the art, opening of a data set is a precursor to accessing the data set, and involves building control structures, locating the data, and otherwise initializing use of the data. In step 502, opening of a data set results in a "call" or invocation of steps 502-505, which prepare for compression according to the invention. Step 503 initialized the counters/indicators also called "wellness variables". Then step 504 returns a message indicating the status of the opening of the data, reporting on the success or failure of the data opening by returning various codes, which may be known to those in the art.

Subsequent to a data set's opening according to step 502-505, the host 102 initiates an input/output operation involving the data set. If the input/output operation involves storage of data in the storage unit 106, a request for compression (a

"compress call") is inherently involved. Namely, the input/output operation constitutes a "calling process", invoking data compression according to the invention. More particularly, task 506 is performed when an individual data block (from an opened data set) is selected from the input buffer 110 for storage in the unit 106, thereby effecting a "compress call" for that data block.

After "compress call" 506, step 508 determines whether the compression evaluation and regulation of the invention (also called compression "wellness") is active. As discussed below, wellness may be deactivated under certain conditions. If wellness is active, control proceeds to step 510 to determine whether a previous compression evaluation has initiated a "skip mode", in which compression is temporarily disabled. This is determined by asking whether the SKIP ACTIVE indicator has been set to a predetermined value such as binary "one". If so, compression in accordance with the present "compress call" will be skipped. In particular, the compression unit 116 in step 512 copies the data block from the input buffer 110 directly to the output buffer 108 without any compression. Then, step 514 increments the NUMBER SKIPPED counter, which represents the number of compress calls skipped in the skip mode. Step 516 then asks whether the NUMBER SKIPPED counter has reached a threshold established by the NUMBER TO SKIP parameter. The NUMBER TO SKIP parameter, preset by a user, is an adjustable parameter that determines how long the skip mode lasts. If the NUMBER SKIPPED counter has reached the NUMBER TO SKIP parameter, the prescribed number of compression calls have been skipped; accordingly, step 518 resets the NUMBER SKIPPED counter to zero, and also resets the SKIP ACTIVE indicator to zero. After step 518 (or a negative answer to step 516), control passes to step 520. Step 520 notifies calling process of the operation results, so that the calling process can take appropriate action as described below. Then, after step 520, the routine 500 ends in step 522.

In contrast to the preceding steps, if step 510 finds that the skip mode is not active (or if wellness is not active), step 526 proceeds with setting up and executing the compression request. In particular, the compression unit 116 in step 526 retrieves the next queued data block from the input buffer 110, compresses the data block, and

stores the compressed data block in the output buffer 108. After step 526, step 528 then asks again whether wellness is active; if not, control passes to steps 520 and 522, where the routine 500 completes. In this case, step 522 signals the calling process to proceed with storage of the compressed data block from the output buffer 108 into the storage unit 106.

If, however, step 528 finds that wellness is active, the routine 500 begins to evaluate the compression performed in step 526. First, in step 530, the compression unit 116 calculates the compression ratio for the compress call satisfied in step 526. As an example, the compression ratio may be calculated according to Equation 1, below.

$$\text{compression ratio} = \text{data size after compression} \div \text{data size before compression}$$

After step 530, the compression unit 116 in step 532 determines whether the compression ratio meets the threshold established by the GOOD COMPRESS RATIO. If so, step 534 increments the GOOD COMPRESS CALLS counter and the step 536 increments NUMBER SAMPLED counter. If the compression ratio falls short of the GOOD COMPRESS RATIO, step 536 merely increments the NUMBER SAMPLED counter.

After step 536, step 538 asks whether the number of samples prescribed by the NUMBER TO SAMPLE parameter has been met. This is determined by asking whether the NUMBER SAMPLED counter has reached the NUMBER TO SAMPLE parameter. If not, the routine 500 ends after completing steps 520 and 522.

When the NUMBER SAMPLED counter reaches NUMBER TO SAMPLE parameter, however, an entire sample set of compress calls is ready to be evaluated. In this case, step 540 asks whether the desired ratio of good samples to bad samples has been met. In particular, the desired ratio has been met if Equation 2 (below) is satisfied.

5

20

## 25

1. GOOD COMPRESS RATIO = 90%.
2. NUMBER TO SAMPLE =  $\frac{16}{64}$
3. GOOD SAMPLE RATIO = 0.75
4. NUMBER TO SKIP =  $\frac{64}{16}$

## 20

25

Various alternative strategies may be used instead, without departing from the scope of the invention. For example, while still using the fixed window, an aggregate compression evaluation may be used instead of evaluating the compression of data blocks individually. With this technique, no single compression call is "good" or "bad". Instead of evaluating the compression of each data block individually, this technique computes a compression ratio by comparing the aggregate size of a sample group of data blocks after compression to their aggregate size before compression.

Another strategy uses a "running window" to evaluate individual data block compression, combining the previously explained approaches. With this technique, each data block's compression is evaluated as performed by the routine 500. However, instead of counting sampling of a predetermined number of compress calls, then determining compression on this fixed window of samples, a "running window" is used. Namely, compression is evaluated after each compress call, by examining the ratio of good/bad compressions in a preceding group of N sample compress calls. After each compress call, the preceding group of N sample compress calls changes, thereby providing a "running window".

Still another approach is to utilize aggregate compression evaluation as discussed above, but with a running window. Certain other variations are also possible, both with the running/fixed window alternatives and the individual/aggregate alternatives. For instance, a running window may contain a set number of compress calls occurring every 2nd, 3rd, 10th, etc. call prior to the current compress call, instead of a preceding group of each N compress calls. For example, the running window may define a group including every 5th call preceding the current call, until a total of twenty calls are found. Likewise, a fixed window may selectively examine compress calls, taking every 2nd, 3rd, 10th, etc. compress call within the prescribed NUMBER TO SAMPLE. Also, instead of ordinal numbers, fixed or running windows may choose some or all sample compress calls using random numbers.

#### Human-Executed Steps: Response to Poor Compression (Figure 6)

As described above in conjunction with Figure 4, appropriate action is taken in step 412 after evaluating one or more compression calls. As one option, task 412 may be performed without human interaction, simply permitting the routine 500 to continue or skip compression as described above.

To boost the efficiency of the compression evaluation and regulation, however, it is advantageous to solicit action from a human operator. Namely, when the routine 500 notifies the calling process of the compression status in step 520, the calling process may assess the situation and decide to deactivate compression, change the

5

10

20

25

18

After the appropriate one of steps 608, 610, or 612 is performed, step 614 restarts the routine 600 under the new mode of operation.

## OTHER EMBODIMENTS

While there have been shown what are presently considered to be preferred embodiments of the invention, it will be apparent to those skilled in the art that various changes and modifications can be made herein without departing from the scope of the invention as defined by the appended claims.

5

2005070599880

19